AR-009-951

DSTO-TR-0461

An Explanation and Analysis Capability for Extended Petri Nets

Mike Davies, Fred D.J. Bowden and John M. Dunn

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

An Explanation and Analysis Capability for Extended Petri Nets

Mike Davies, Fred D.J. Bowden and John M. Dunn

Information Technology Division
Electronics and Surveillance Research Laboratory

DSTO-TR-0461

ABSTRACT

Extended Petri nets (EPN) have been adopted to help establish a number of capabilities for the study of military Command, Control, Communication and Intelligence (C3I). Regardless of the specific application, an EPN explanation and analysis capability is very important. This is particularly true where EPN have been used to describe artificial representations of real-world C3I entities. Some explanation and analysis capability is needed so that the underlying assumptions and characteristics of such an artificial representation can be clearly conveyed to, and judged by, an appropriate military domain expert. This document reports a general-purpose explanation and analysis capability for a class of EPN, that has been developed as part of the Distributed Interactive C3I Effectiveness (DICE) simulation activity within Information Technology Division.

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

Published by

DSTO Electronics and Surveillance Research Laboratory
PO Box 1500
Salisbury South Australia 5108

Telephone: (08) 259 5555
Fax: (08) 259 6567

AR No. 009-951
December 1996

APPROVED FOR PUBLIC RELEASE

An Explanation and Analysis Capability for Extended Petri Nets

Executive Summary

This report details an explanation and analysis capability for extended Petri nets (EPN) that helps convey clearly the underlying characteristics and assumptions of an EPN. The capability is written using the programming language Prolog and is general-purpose in that it is not dependent on a particular application of EPN. The power of such a capability becomes evident when an EPN is used to represent for analysis purposes the features of some existing real-world system. Prior to conducting the analysis it is vital that the EPN representation matches adequately the real system. The closeness of such a match can generally only be judged by some domain expert who may or may not be familiar with the language associated with Petri nets.

The EPN explanation and analysis capability accommodates such queries as:

- What are the possible sequence of actions that could occur from some initial state to some final state? How long would each sequence take and with what probability?
- Why did or would a certain action occur?

The results from such queries are both abstracted to a user-defined level of detail and presented in a easily understandable textual form.

Information Technology Division (ITD) of the Defence Science and Technology Organisation was tasked by the Headquarters Australian Defence Force to develop, through modelling and simulation, tools for effectiveness studies of Command, Control, Communication and Intelligence (C3I) systems. EPN and their simulation are being used to provide both a stand-alone system analysis capability and as computer-based representations in an interactive simulation. The explanation and analysis capability reported in this document is developed to ease analysis, judgement and modification of EPN characteristics by military domain experts. To facilitate the use of this capability, a graphical user interface environment has been developed.

Authors

Mike Davies
Information Technology Division

Mike is a senior research scientist with a BSc(Hon) in applied mathematics and a PhD in mathematical modelling. His career at the DSTO has involved the application of mathematical modelling and computer simulation to the analysis of human performance in ground-based air defence systems and aircraft navigation and weapon delivery. His current research interests concern the evaluation and enhancement of military Command, Control, Communication and Intelligence through simulation-based analysis and training.

Fred D.J. Bowden
Information Technology Division

Fred Bowden completed his Bachelor of Science at Murdoch University in 1989 majoring in Mathematics and Physics. He joined Combat Systems Division (now part of Information Technology Division) in 1990. In 1993 Fred completed a First Class Honours degree in Applied Mathematics at the University of Adelaide. He is currently studying for a doctorate relating to the application of extended Petri nets to military Command, Control, Communications and Intelligence systems.

John M. Dunn
Information Technology Division

John Dunn completed a Bachelor of Applied Science degree in Mathematics and Computing at the South Australian Institute of Technology in 1988. He has worked at DSTO for seven years and is a Professional Officer primarily involved in programming and computer support on PC and UNIX based systems.

Contents

Figures

1. Introduction

Information Technology Division (ITD) of the Defence Science and Technology Organisation was tasked by the Headquarters Australian Defence Force to develop, through modelling and simulation, tools for effectiveness studies of Command, Control, Communication and Intelligence (C3I) systems. The primary tool developed is the Distributed Interactive C3I Effectiveness (DICE) simulation[1] that will permit a thorough examination of current and proposed C3I-related procedures and technologies within the framework of simulated military activities, eg missions, operations or battles.

The technique of extended Petri nets (EPN) has been used by many researchers in a variety of applications that include the study of C3I systems. The use of EPN, and in particular their implementation as simulations, has been adopted by ITD to establish a number of capabilities that permit the study of military C3I, the main ones being:

- Description and implementation of artificial players, or *agents*, in the DICE simulation that complement and communicate with the human players that interact with the simulation[1]. An agent might represent a centre of decision making; information processing or filtering; information transfer; or combinations of these.

- Provision of a non-interactive C3I systems analysis capability that enables inspection of general system characteristics without the constraints of an interactive environment.

Whatever the application, a capability to analyse the EPN under simulation is very important. This is particularly true where EPN have been used to establish DICE simulation agents, in which case the underlying assumptions need to be easily and clearly conveyable to an analyst or to a military person assessing the credibility of the artificial representation. A similar capability is important when describing a C3I system.

This document reports a general-purpose explanation and analysis capability for a class of EPN, written using the declarative language Prolog. It should be noted that the actual EPN simulations, mentioned above, are not discussed in this report. The use of Petri nets is part of ongoing research into the applicability of this technique to the study of C3I.

2. Extended Petri Nets

Petri nets (PN) were originally developed by Carl Petri as a means of modelling computer systems. Since their original conception they have been used to model and analyse many different types of systems. PN are ideal for the modelling of event systems which involve the problems of resource sharing, concurrent and sequential processes, and synchronisation. The main problem with PN is the growth in model size as the systems they represent become complex. To overcome this problem PN have been extended by the introduction of features such as coloured tokens, firing modes and hierarchies. These extensions allow the user to represent more complex systems in a manageable way; however, they do not increase the modelling power of PN.

The original work carried out by Petri and many other researchers using PN as a modelling tool, involved structural analysis of systems, such as searching for deadlocks and invariants. The PN definition used by Petri did not include time. However, PN have been extended to include time, which allows for performance studies based on such features as mean waiting times and throughput. The introduction of time also allows PN to be used as a modelling tool in interactive simulations.

2.1 Some Definitions

An EPN is a directed graph with two types of nodes: *places* and *transitions*[7]. Pictorially, places are indicated by circles or ellipses and represent entities such as conditions and buffers. Transitions are displayed on the graph as rectangles or bars and represent concepts in the real system such as processes, algorithms, and events. The nodes are joined by one of two types of directed arcs: *input arcs* and *output arcs*. An input arc goes from a place to a transition and an output arc runs from a transition to a place. Each transition has corresponding sets of *input places* and *output places*. It should be noted that arcs can only go from a place to a transition or vice versa. *Tokens* make up the final elements in an EPN and their positions define the state of the system and describes situations such as availability of resources, satisfied conditions and items in a buffer. Tokens are represented graphically by identical dots and, in the case of PN, can only be found in places. Coloured PN extend this by allowing tokens to be differentiated between; that is, introduce *coloured tokens* into the PN model. The movement of tokens between places is controlled by the transitions of the PN. Input and output arcs are labelled with coloured tokens to signify the input requirements and output characteristics of each transition.

A transition which has its input requirements satisfied is said to be *enabled*. When an enabled transition is activated, the marking changes and it is said to *fire*. Upon firing, the transition removes the specified tokens from its input places and deposits the relevant tokens in its output places as defined by the inscriptions on the arcs. A transition can have a number of *modes* of firing, each with different input and output features. This will be introduced later; until then each transition can be regarded as having one and only one mode denoted by *fj-k* to signify mode *k* of transition *j*.

As an example, consider the EPN in Figure 1. There is no initial marking indicated visually but assume there are two c1 tokens in place p1 and a single c2 token in place p4. Modes f1-1, f2-1, f3-1 and f4-1 of transitions t1, t2, t3 and t4, respectively, are enabled. If mode f1-1 fires then two c1 tokens are removed from place p1 and two c1 tokens are placed in place p2, changing the marking to two c1 tokens in place p2 plus the remaining c2 token in place p4. This results in modes f3-1, f4-1 and f5-1 being enabled.

Time can be introduced into PN in a number of different ways[7]; in this case time is considered to be represented as *enabling times*. That is, once a transition's mode of firing becomes enabled it must remain enabled for a specified time before it can fire. This firing time can be deterministic or stochastic. Both timed and *immediate* (ie zero firing time) modes can be accommodated which provides a powerful modelling capability. Immediate modes have an associated weighting which is used in the resolution of conflict as discussed below.

2.2 Execution Strategy

Execution strategies are employed in conflict resolution and a number of different strategies can be applied[7]. A specific strategy has been adopted within the explanation and analysis capability and is discussed here. A conflict is said to occur between modes for a given marking if more than one mode is enabled at the same time and the firing of any one of these will disable the remaining enabled modes. In Figure 1, conflict occurs between mode f3-1 and f4-1 when there is only one c2 token in place p4. The following strategy is a combination of race and preselection strategies defined for generalised stochastic Petri nets[7].

(i) If all the enabled modes are timed then the mode with the shortest time fires. (Sampling would be used in the case of stochastic firing times.)

(ii) If all the enabled modes are immediate, the weightings of each are used to determine, probabilistically, which one will fire. If the weighting of a transition t is denoted by $\Omega(t)$, and the sum of the weights of all the enabled immediate modes is denoted by X, each transition t will fire with probability $\Omega(t)/X$. (A more detailed account of this process is given in Appendix A.)

(iii) If a combination of immediate and timed modes are enabled then only the immediate modes are considered and dealt with according to rule (ii).

2.3 Hierarchical Extended Petri Nets

The introduction of hierarchies into EPN has allowed the modelling of even more complex systems as now the designer can work at a level of detail required for the development being carried out. In the application reported here, hierarchies are introduced into the EPN through the use of substitution transitions. A substitution transition is a transition corresponding to an entire EPN that represents the detailed processes of the event the transition represents. In Section 3, an example of how hierarchical EPN are constructed and interpreted is given.

3. A Role-Based Methodology For Extended Petri Nets

A role-based methodology has been developed for EPN and research is ongoing into the applicability of this approach to representing C3I systems and agents that exhibit C3I characteristics. The role-based methodology facilitates EPN design and construction by permitting a modular structure. The use of roles also allows the formation of hierarchical EPN and information abstraction which form the key to the explanation capability detailed later. A library of re-useable roles, where each role performs a certain function, can be created and accessed, permitting a building block approach in the creation of an EPN. A brief introduction to the relevant aspects of the methodology follows[3].

The systems being modelled by the technique described in this document have a natural breakdown of their processes into smaller components. These components will be loosely termed "roles". A role is not necessarily the lowest-level or base activity of the system, it is more a function which may in turn have many roles within it. This structure leads to a hierarchical design technique. By taking a hierarchical approach, the complexity at the level the designer is working at is reduced to a level which is relevant for the work being conducted. This method also supports the natural break down of complex systems into simpler ones. This means the system's roles can be designed separately and then brought together to form the overall model. Alternatively the designer can initially sketch out the basic functions of the system and then add detail by instantiating its roles independently. Both a top-down and bottom-up design and implementation capability therefore exists. The system being modelled and the information about the system will determine which method is most appropriate for the task at hand. This method also makes changes to the model easier as they will only involve changing a limited number of roles.

The simplest way of introducing the relevant methodology is through the use of an example. Consider the situation where a pilot requests permission from an air traffic control tower to taxi to a runway so that he can begin preparation for take off. (For convenience the authority shall be referred to as ground control.) If permission is given then the pilot will proceed along the taxiway, otherwise he waits for a period of time before submitting the request again. If an aircraft is taxiing, then the taxiway is considered unavailable to other aircraft. On receiving a request, ground control must check both the current weather conditions and the availability of the taxiway before deciding if the pilot is cleared to taxi.

Figures 2 to 5 show the role-based model representation of this system along with descriptions of the significance of tokens residing at particular places in the nets. In Figure 2, the higher level model corresponding to the pilot level is shown. First the pilot submits his request, which is represented in the EPN by the firing of transition t1 and the creation of a c2 token in place p2. Ground control then responds, either approving or disapproving the request, ie substitution transition T1 fires. If permission to taxi is denied then a c1 token is created in p1 after which the process continues, with the pilot waiting before submitting a new request. If the request is

approved then this is signified by a c3 token created in p3 following which the pilot proceeds to taxi. Transition t2 fires on completion of the taxiing placing a c4 token in p4 to indicate the pilot has completed taxiing and a c5 token in p5 signifying that the taxiway is now available.

In Figure 3 the details of ground control are modelled. The extended PN shown here is that associated with substitution transition T1 in the PN of Figure 2. This model involves two different roles; the checking of the weather conditions (substitution transition T1) and taxiway availability (substitution transition T2). Permission to taxi is granted (token c7 in place p5) if the weather conditions are satisfactory and also the taxiway is available. If either of these requirements are not met then permission is denied, taking the form of a c8 token in place p6.

The substitution transitions T1 and T2 correspond to the EPN shown in Figures 4 and 5 respectively. The form of these nets corresponds with the connectivity associated with T1 and T2 in the ground control EPN. The important feature to note here, however, is that the nets of Figures 4 and 5 are general-purpose in nature in that the actual conditions and resource checked are not specified. It is only when the nets become instantiated as roles within the ground control net that the conditions become the weather and the resource is stipulated as being the taxiway.

It is important to note that with the type of hierarchy used, all the EPN must be EPN in their own right. That is, they must be complete, in that any input and output arcs of the substitution transitions must be correctly labelled. In some definitions of hierarchical PN, this is not required as it is considered unnecessary as the semantics of the PN around this transition are represented at the substitution transition level. However, in our case due to the explanation and other requirements the higher level EPN must also be correctly labelled. Using this example, the terminology used will be introduced.

When the transition of an EPN is defined as a substitution transition, a subnet is defined to represent the detail of the transition. That is, a role is introduced to represent the transition. In defining subnets for a net, the input and output place of the substitution transition, called the *socket places*, are assigned corresponding places in the subnet, called *port places*. It may be that for a particular application the relationship between socket and port places is not one to one. In addition to this some port places may be unattached for a given representation if the tokens associated with that place are not needed. Consider the example in Figures 2 and 3 where T1 is a substitution transition. The mapping of pilot net socket places to ground control port places is given below:

Socket Place (Pilot)	Port Place (Ground control)
p1	p6
p2	p1
p3	p5

When the substitution transition is linked to the subnet representing it, unique identities must be assigned to token colours, places and transitions to avoid clashes. Examples of such assignment are given in Section 6.

As set out earlier the inscriptions of all the nets representing the agent must be complete, this includes those with substitution transitions. This also means that substitution transitions have firing modes, even if they are only nominal. These firing modes are called *surface modes*. The EPN associated with the substitution transition is termed the *role net*. Thus, the user may refer to the surface modes of a net without having to consider what is happening within the subnets of that net. For the substitution transition in Figure 2 there are two surface modes of firing, one corresponding to each of the possible outcomes of the instantiated role; ie the permission to taxi is given or denied.

4. Requirements Of the Explanation And Analysis Capability

In our study of military C3I, EPN are translated into computer simulations that are employed to represent DICE simulation artificial agents[1,7] and provide tools for C3I systems analysis. An explanation and analysis capability is required whereby the underlying assumptions and characteristics of an EPN can be inspected and clearly conveyed. This capability would facilitate judgement by a military domain expert of the credibility and realism of an EPN simulation against the real system that that EPN is meant to describe. Presentation of the EPN analysis system and the best way of displaying information returned by such a system is not a trivial problem; this report is concerned with the provision of such information. The ability to have agents, for example, explain their actions in an easily understandable form, is recognised as a very important requirement in the Distributed Interactive Simulation (DIS) arena[2,6].

A number of required features were identified and these are given in the following sections.

4.1. Reaction to a given situation

The reaction or response to a given situation is considered to be the *outcomes* that eventuate as a direct consequence of the situation or state. In this case, an outcome can take the form of one or more events, referred to as *actions*, where the multiplicity will result if the actions have occurred simultaneously. In some circumstances, one or

more of a number of outcomes might occur as a result of a given situation. In such a case, each possibility should be indicated by the analysis facility, along with an associated likelihood of each occurring. Each action within an outcome should also, if appropriate, be accompanied with the time at which it would occur relative to that of the given situation. The situation that eventuates as a consequence of an individual action, or the outcome overall, should also be obtainable.

4.2. Sequence of actions resulting from a given situation

An important capability is to generate outcomes that consist of a time-sequenced succession of actions that could result from a given situation. One method of achieving this is to incrementally step through the reactions that result from the given situation through successive uses of the capability described in Section 4.1. If an incremental approach were undesirable then there would need to be some way of telling the analysis system how far ahead to generate the actions. One method would be to have the system generate a sequence of actions that starts at the initial situation and ends at some user-specified situation (or at a situation which has some user-specified portion). Again, if appropriate, time and probability data should accompany the information.

4.3. Explanation of actions

An important requirement is the ability to explain why a certain action occurred (or would occur). This might be addressed with a description of the situation that caused that particular action, or it may be more appropriate or meaningful to inspect the preceding actions leading up to the action under interrogation.

4.4 Sensitivity of actions that could arise from a given situation

This requirement refers to the ability to inspect the sensitivity of actions to a specified situation. Such a capability would indicate to what extent the occurrence of an action was dependent on a given situation, and to what extent that situation would have to change before an action would or would not occur. This is comparable with a facility reported in reference 2 that inspects what would happen if a situation were slightly different to that previously used for analysis.

4.5 Actions that could result in a given situation

This requirement addresses the ability to inspect what action or combination of actions could cause a certain situation to arise. This is almost the reverse of capability discussed in Section 4.2 in that the system is working backwards from a given situation rather than forwards.

4.6 Actions that could contribute to a given situation

This requirement is identical to that mentioned in Section 4.5 except that the resulting actions do not have to produce the given situation exactly but only have to contribute to it in some way. This requirement is also similar to that discussed in Section 4.4 in that what needs to be generated is the sensitivity of action output to a given situation.

4.7 Information abstraction

When utilising the capabilities discussed in the preceding sections, it can be expected that, in some cases, the information generated by the analysis system would be considerably detailed. Information should be presented, however, at a level of detail commensurate with the interests of the analyst that uses the system. Some summary or process of abstraction should be employed to achieve the appropriate level. The analyst should, therefore, be given the capability to indicate the appropriate level and to change this level when necessary to obtain more or less information during any investigations.

5. Representation Of Extended Petri Nets: Basic Properties

The declarative language Prolog[4,5] is a particularly suitable language for describing EPN, where the basic connectivity characteristics can be regarded as a set of facts or procedures plus logical conditions. Perhaps the most powerful feature of Prolog, with regard to establishing an explanation and analysis capability is its ability to represent search and multiple solution identification. Furthermore, Prolog is an ideal language for conducting, for example, natural language processing; and message translation and filtering which may be important future features of agents in the DICE simulation. The following sections detail representation of EPN and the implementation of other important attributes associated with EPN, using Quintus Prolog Version 3.1.4[5].

5.1 Notation and basic connectivity

The Prolog version of the sample EPN in Figure 1 is shown Figure 6. The procedures shown form the database file which contains specific details concerning the Petri net under inspection. This database file is utilised by the EPN explanation and analysis programme, which is general purpose in nature.

The notation used in the diagrammatical representation of an EPN is given in Section 2. In the Prolog representation, place, transition, transition mode and token identities can take any form starting with a lower-case letter. In this example, places, transitions and token colours are given unique identities of the form p*i*, t*i* and c*i* respectively, where *i* can take any positive integer value.

Being an EPN, transitions can have a number of modes of firing. The *modes* procedure gives the list of transition modes that apply to the EPN concerned. Transition modes are considered to take the form f*i*-*j*, signifying mode *j* of transition t*i*. In the example of Figure 6, each transition has only one mode of firing. Although this is a simplification diagrammatically, it is not in terms of the Prolog programme since each mode is considered individually regardless of its association with a particular transition.

Generally, places can hold any number of tokens of more than one type and hence the enabling of transition modes is determined by consideration of the distribution and quantity of such tokens. Transition mode firings result in the re-distribution of such

tokens. The *trans_mode_input* procedures describe the input requirements of each transition mode. There is one procedure for each of the modes listed in the *modes* procedure. The input requirements of a mode take the form:

[[p3,c1,2], [p5,c3,2]]

which indicates the location, colour, and multiplicity required of the token types concerned. Each element within the input requirements (eg [p3,c1,2]) is referred to as an *input requirements component*. (Elements of the form [p6,c8,9] are generally referred to as *components*.) In this example, two tokens of type c1 are required at place p3 in addition to two tokens of type c3 at place p5 in order to enable the mode concerned.

The *trans_mode_output* procedures describe the output features of each mode. There is one procedure for each of the transitions listed in the *modes* procedure. The output requirements of a transition mode takes an identical form to that of the *trans_mode_input* procedure. That is, a list indicating the token distribution that results as a consequence of firing the mode concerned.

5.2 Weights and deterministic and stochastic firing times

Also contained in the database section are procedures that describe the firing time characteristics of the transition modes, as shown in Figure 6. Immediate and timed modes can be represented in the programme database through the *trans_mode* procedure which takes the form:

trans_mode(Net, Mode_Id, Mode_Description, Weight_or_SD, MeanFiringTime).

The first three parameters indicate the associated net, mode identity and firing description respectively. Parameters 1 and 3 are used primarily in the provision of the explanation capability discussed later. Parameters 4 and 5 are used to specify firing attributes of the mode and to signify the mode type. For example, in Figure 6, mode f3-1 is described by:

trans_mode(petri_net, f3-1, 'Mode f3-1 - an immediate mode', 2, 0).

where parameter 4 indicates the weight for the immediate mode and in parameter 5 is contained a zero which signifies that the mode is immediate. A timed mode would take the form:

trans_mode(petri_net, f4-1, 'Mode f4-1 - a timed mode', 0, 8.2).

where the timed nature is indicated by a non-zero value in parameter 5 which gives the mean firing time of the mode. A zero value for parameter 4 signifies, as in this example, that the firing time is deterministic. A non-zero value for this parameter would indicate that the firing time is stochastic and could also be used, if required, to indicate the measure of dispersion in the firing time.

It should be noted that analysis of stochastic transient features of EPN is the subject of doctorate research being conducted by one of the authors (FDJB) and will be reported during the course of, and following, that research. This report will be concerned with stochastic features of EPN but only those involving immediate transition modes.

5.3 Enabling modes

Enabled modes are those whose input requirements are satisfied by the current marking of the EPN. A marking takes the form:

[[p1,c1,2], [p3,c2,1], [p3,c1,3]]

which reads in a similar fashion to the input and output associated with a mode, as discussed in Section 5.1. For the purpose of the Prolog software, a valid marking has at most one component for a given place and token type combination and the number of tokens for that component is greater than zero.

Determination of enabled modes can be best illustrated through the use of the *response* procedure which determines those modes within the net that are enabled by a given marking. For example, using the net depicted in Figure 6, the goal:

response([[p1,c1,5]], FinalMarking, Firings, EnabledModes, Probability, Time).

would result in:

EnabledModes = [f1-1, f2-1].

5.4 Firing modes

Consideration of which modes to fire needs to be made whenever more than one mode is enabled. Although more than one mode can be considered to fire simultaneously, this is only possible if timing considerations allow and also if the initial marking of the EPN can supply adequately numbered and distributed tokens to meet the demands of the firings. Timing considerations are addressed in the execution strategy employed, whilst the latter consideration of token supply versus demand is dealt with by resolving any conflict that exists between the demands of the modes.

The execution strategy for EPN is discussed in Section 2.2. The general policy is that the mode with the least firing time is fired. Immediate modes therefore have priority over timed modes. Furthermore, at most one timed mode can fire at any one time since this individual firing might consequently affect the firing of other currently enabled timed modes by the time their instant of firing arrives. The Prolog analysis programme assumes, however, that any one of the modes could eventuate as the one with the least firing time. Each of the possible solutions would have some corresponding likelihood.

A conflict is said to occur between modes if more than one mode is enabled at the same time and the firing of any one will disable the others. Owing to the execution strategy, only enabled immediate modes need be inspected for confliction. Considering the example in Figure 6, if a marking of [[p2,c1,2], [p3,c1,2], [p5,c3,3]] is assumed then a condition of conflict exists since each of the modes f5-1, f6-1 and f7-1 are enabled but not all can fire. That is, each mode is demanding tokens from within the marking and there are not enough tokens to satisfy this demand. For example, firing f6-1 will disable transition modes f5-1 and f7-1. In the Prolog analysis facility all possible outcomes must be generated and considered as alternative solutions, with an associated likelihood, to the confliction problem.

Once modes have been selected for firing the actual firings involve simply the distribution of tokens according to the output characteristics of the modes concerned. Adoption of the execution strategy and the mechanism for resolving conflict can be best illustrated through the use of the *response* procedure. For example, the goal:

response([[p2,c1,2], [p3,c1,2], [p5,c3,3]], FinalMarking, Firings, EnabledModes, Probability, Time).

would result in:

EnabledModes = [f5-1, f6-1, f7-1];

Firings = [f6-1];

FinalMarking = [[p2,c1,1], [p3,c1,1], [p6,c4,1]];

Probability = 0.167;

Time = 0.0;

or (with the same enabled modes)

Firings = [f5-1, f7-1];

FinalMarking = [[p5,c3,1], [p6,c4,2]];

Probability = 0.833;

Time = 0.0.

All of the enabled modes are immediate and hence are all inspected for conflict and the possible outcomes identified. Calculation of the associated occurrence probability of each of the outcomes is discussed in Section 5.5 and detailed further in Appendix A.

The goal:

response([[p4,c2,2]], FinalMarking, Firings, EnabledModes, Probability, Time).

would result in:

EnabledModes = [f3-1, f4-1];

Firings = [f3-1];

FinalMarking = [[p3,c2,2], [p4,c2,1], [p5,c3,1]];

Probability = 1.0;

Time = 0.0;

where the enabled modes consist of one timed and one immediate mode, resulting in the immediate mode being inspected for confliction and subsequently firing. The timed mode was disregarded despite there being enough c2 tokens at place p4 to fire both f3-1 and f4-1.

The goal:

response([[p1,c1,5]], FinalMarking, Firings, EnabledModes, Probability, Time).

would result in:

EnabledModes = [f1-1, f2-1];

Firings = [f1-1];

FinalMarking = [[p1,c1,3], [p2,c1,2]];

Probability = 1.0;

Time = 3.2

where the enabled modes consist of two timed modes. One solution to the response procedure with probability of unity indicates that f1-1 will always fire prior to f2-1.

5.5 Firing probabilities

As illustrated in Section 5.4, the Prolog analysis programme identifies all the possible outcomes that could occur as a result of a given marking. Numerous outcomes could occur as a consequence of firing time considerations and conflict resolution. Each outcome will have some associated likelihood or probability which will be dependent on the process by which the firings are selected in the EPN simulation under inspection. This particular aspect of the execution strategy is a topic of ongoing research in the case of timed transitions. In Appendix A is detailed a general strategy that has been employed and incorporated in Prolog for immediate transitions where firings are determined by sampling over a uniform distribution characterised by the mode weights.

5.6 Representing roles

In discussing the implementation of EPN roles, the example given in Figures 2 to 5 is used. For clarity, it is advantageous to consider again the building of this example, as discussed in Section 3.1.

The Check Conditions and Check Availability of Resource nets would initially be described diagrammatically as shown in Figures 4 and 5. The Prolog EPN equivalents of these nets are given in Figures 7 and 8 respectively. The form of the Prolog procedures in these figures is identical to that of Figure 6 and discussed in earlier sections. There is one additional procedure, however, that has been introduced at this stage and that is the *place_token* procedure. This procedure gives a description of the

significance of a particular token residing at a particular place in the net and is an important procedure in the provision of the analysis capability discussed later.

Consider now using these nets as roles within a Ground Control net as shown in Figure 3. When a role is *instantiated* as a subnet within a given net, it is assumed by the Prolog software that the following will be achieved, through a combination of manual and automatic procedures:

(i) Given that a net may be used to provide a number of roles, an instantiation of a role is given a unique name that describes the manner in which the associated net is being used in that particular instantiation.

(ii) Mode (including any surface modes), place and token and identities associated with the *role net* are made unique by appending their original forms with the role name. This applies to roles within the newly formed role, including the role names of those roles.

(iii) All port places associated with a role are connected to the appropriate socket places of the net for which that role is a subnet; token equivalence is also established.

(iv) Fusion places are recognised.

(iv) Each mode of the role net that can distribute tokens to socket places, and hence influence the net for which that role is a subnet, is summarised by a surface mode.

During the instantiation process, the user might have the power to change any of the information associated with the role net (and any roles within it), with the exception of the fundamental connectivity and mode input and output characteristics. (Changes to these latter features necessitate creation of a new role net.) That is, the user can alter the firing time (mean firing time and statistical data), and descriptive information associated with the role net and any roles within it. This allows the net to be tailored to each instantiation.

The Prolog EPN representation of the resultant architecture is one where the architecture has been essentially exploded and treated, in the manner described in earlier sections, as one net. For the example considered, assume that the Check Conditions and Check Availability of Resource nets have been instantiated into the Checking Weather (role name: checking_weather) and Checking Taxiway (role name: checking_taxiway) roles. The resultant exploded Ground Control net is given in Figure 9.

It can be seen that the equivalence of the socket and port places is recognised and only the socket places are considered. In this case, all the places associated with the role nets were port places. The layered attributes of the role-based architecture are signified, however, through use of the *surface_mode* procedures. Surface modes are symbolised in the form fs2-1 indicating surface mode 1 of role 2. Each of the modes belonging to the Check Conditions and Check Availability of Resource nets, upon firing, distribute tokens to the Ground Control net and can therefore influence that net.

For that reason, each of these modes is summarised by a surface mode through the use of a *surface_mode* procedure. The *surface_mode* procedure states the net to which the surface mode belongs (in this case, they all belong to the Ground Control (gc) net); the original or regular mode that the surface mode summarises; a description of the surface mode; and the net to which the regular mode belongs. It is the *surface_mode* procedures alone, then, that characterise the role-based nature of the eventual architecture. In the instantiations of the Check Conditions and Check Availability of Resource nets, it has been assumed (as it will in other instantiations in this report) that mode firing time information is not changed and also that descriptive information has changed to the extent that it gets tagged with the role name in a similar manner to that stated in item (ii) above. For the purposes of the examples in this report, the latter is a simple systematic way of tailoring descriptions to particular instantiations.

Similar steps are taken in instantiating the Ground Control net as a role (role name: gc) within the Pilot net. In this case, places p1, p5 and p6 of the former net are considered as port places and Pilot net surface modes are described that summarise the Ground Control net surface modes fs1-2, fs2-1 and fs2-2 since these are the only ones that distribute, upon firing, to the port places. Places p2, p3 and p4 of the Ground Control net remain in the instantiated role net and consequently place/token descriptions for these places remain unchanged but are considered tailored by being tagged with the role name 'gc'. Place p5 in the Pilot net is a fusion place with place p4 in the Ground Control net. In the exploded net, only one of these places is used and, in this example, it is assumed to be that of the Ground Control net. The resultant exploded Pilot net is given in Figure 10 and is used considerably to illustrate the explanation and analysis capability addressed in Section 6.

A similar procedure would be carried out in the creation of the architecture displayed in Figure 11. In this case a generic Controller net is employed which is similar to the Ground Control net in that it uses both the Check Conditions and Check Availability of Resource nets as roles. However, in this case the Controller net is considered to be a general-purpose controller that checks some condition and some resource. The representation of this net is given in Figure 11. The Controller net gets instantiated twice, as the Ground Control (T1) and Air Control (T2) roles, in forming the Pilot net for this example. It should be noted that specification of what condition (weather (p5)) and what resources (taxiway (p6) and airway (p10)) is done at the Pilot net level. The weather is checked by both ground and air control. Hence, places p2 and p4 of the Controller net (see Figure 3 for structure) have, in this case, been used as port places which, upon instantiation, connect to socket places p5, p6 and p10 within the Pilot net. The corresponding Prolog representation for this architecture is given in Figure 12.

6. Explanation And Analysis Capability

The examples shown in Figures 2 to 5 and 11, with associated Prolog database representations in Figures 7 to 10 and 12, will be used to illustrate the EPN explanation and analysis capability in some detail. In Section 4 were presented a number of key requirements of an EPN explanation and analysis capability and these are addressed

individually in the following sections. It should be noted that information abstraction features are illustrated and discussed throughout the majority of the following sections. Information abstraction is generally achieved to a user-specified level (the *target net*) that corresponds to an appropriate level in the EPN hierarchy. Prolog procedures are introduced when needed, full details being found in reference 9.

6.1 Reaction to a given situation

In an EPN, a situation is described by a certain marking; the form of a marking and its representation in the Prolog programme is discussed in Section 5.3. A useful procedure is one which takes a given marking and produces some meaningful description of the associated situation. This capability is provided by the *situation* procedure. For example, assuming the EPN given in Figures 2 through 5 and 10 (this net will be used for all examples unless stated otherwise), the goal:

situation([[p2,c2,1], [gc_p3,gc_c4,1]], Description).

would result in:

Description = ['Permission to taxi requested', 'gc: Weather allows taxi']

which is derived directly from the place/token combination descriptions specified through the *place_token* procedures. It should be noted that information abstraction is not applied to the output of this procedure; the procedure returns a description of the complete marking, regardless of the particular location of tokens. In the case of the goal:

situation([[p2,c2,3]], Description).

Description = ['Permission to taxi requested'+3]

signifying the multiplicity of the token c2.

The ability to determine the reaction to a given situation is provided by the *desc_response* procedure which is the previously discussed *response* procedure with an added description and information abstraction capability. For example, for the same EPN, the goal:

desc_response(pilot, [[p2,c2,1], [gc_p2,gc_c3,1]], Outcome, ResultantSituation,

Probability, Time, NewMarking, FullFirings, RawResult, _).

requires the response to the initial marking [[p2,c2,1], [gc_p2,gc_c3,1]] (ie ['Permission to taxi requested', 'gc: Weather not suitable for taxi']) with information being abstracted to the Pilot net level (the *target net*). This goal would return:

Outcome = ['Ground Control determined that permission should not be

granted for taxi'];

ResultantSituation = ['gc: Weather not suitable for taxi', 'Taxi not allowed'];

Probability = 1.0;

Time = 6.0;

NewMarking = [[gc_p2,gc_c3,1], [p1,c1,1]];

FullFirings = [gc-checking_weather_f1-2];

RawResult = [fs1-2];

The mode that has actually fired in response to this goal is gc-checking_weather_f1-2. The *surface_mode* procedure has been used to identify that this mode corresponds to the surface mode gc_fs1-2 in the Ground Control net and subsequently that this surface mode corresponds to fs1-2 in the Pilot net which is the net under consideration. The Pilot net surface mode is therefore given as the raw outcome (RawResult) and this is translated into a descriptive form (Outcome). Information abstraction has therefore been achieved in returning the outcome to the stated situation.

Consider the goal:

desc_response(gc, [[p1,c1,1], [gc_p3,gc_c4,1], [gc_p4,gc_c6,1]], Outcome,

ResultantSituation, Probability, Time, NewMarking,

FullFirings, RawResult, _).

that is, the target net is now at the Ground Control level and full details of the firings are not generated. This would return:

Outcome = ['gc: Determined that taxiway is not available for taxi'];

ResultantSituation = ['Taxi not allowed'+2,'gc: Taxiway not available'];

Probability = 1.0;

Time = 4.0;

NewMarking = [[p1,c1,2],[gc_p4,gc_c6,1]];

FullFirings = [gc-checking_taxiway_f1-2];

RawResult = [gc_fs2-2].

Abstraction has now stopped at the Ground Control net. It should also be noted that f1-1 did not fire owing to gc-checking_taxiway_f1-2 having a shorter firing time.

Another useful example is the goal:

desc_response(pilot, [[p2,c2,1], [gc_p2,gc_c2,1]], Outcome, _, Probability, Time,

NewMarking, FullFirings, RawResult, _).

which would return:

Outcome = [];

Probability = 1.0;

Time = 6.0;

NewMarking = [[gc_p2,gc_c2,1], [gc_p3,gc_c4,1]];

FullFirings = [gc-checking_weather_f1-1];

RawResult = [].

That is, as far as the Pilot net is concerned, there is no response to the initial marking. Changing the target net to the Ground Controller would identify the single firing within that net. Alternatively, one of many utilities, reported later, could be used to inspect the sensitivity of modes to the initial marking or sequences of firings that eventuate from the initial marking, for example. (A description of the resultant situation was not requested in this example.)

The above examples illustrate that the presence of roles allows explanations of outcomes (and the actions within) to be presented at a level appropriate to a user's query; that is, maybe at a significant level of abstraction from details of actions that result in low level roles of the scenario.

If more than one outcome could arise from an initial situation, the *desc_response* procedure would identify these as multiple solutions in an identical manner to that described for the *response* procedure in Section 5.4. The *desc_response* procedure could be used successively to incrementally generate a sequence of actions that occur as a result of some initial situation.

6.2 Sequence of actions resulting from a given situation

The *marking_sequence* procedure takes some initial marking and some final marking and determines the sequence of mode firings, and accompanying markings, that are required in order to reach the final marking from the initial one. The procedure uses the principle of depth-first search. Multiple solutions to the marking and firing sequence problem are capable of being generated by this procedure. The powerful features of Prolog allow partial specification of the final marking.

The capability of the *marking_sequence* procedure is used by the *sequence_of_actions* procedure. This procedure generates an outcome consisting of a sequence of actions that occurs in response to a stated initial situation and resulting in some fully or partly

defined final situation, and also provides the necessary descriptive features and information abstraction typified in Section 6.1.

For example, the goal:

sequence_of_actions(pilot, [[p1,c1,1], [gc_p2,gc_c2,1], [gc_p4,gc_c5,1]], [[p4,c4,1]|_],

MarkingSequence, FullFiringSequence,

AbstractedFiringSequence, Outcome, Probability, Time, _).

requires generation of marking and firing sequences from an initial marking:

[[p1,c1,1], [gc_p2,gc_c2,1], [gc_p4,gc_c5,1]]

that signifies the situation:

['Taxi not allowed', 'gc: Weather suitable for taxi', 'gc: Taxiway available']

to a final marking:

[[p4,c4,1]|_]

that signifies any situation which contains the state:

['At runway'] .

The response to this goal would be:

MarkingSequence = [[[p1,c1,1], [gc_p2,gc_c2,1], [gc_p4,gc_c5,1]],

[[gc_p4,gc_c5,1], [gc_p2,gc_c2,1], [p2,c2,1]],

[[gc_p4,gc_c5,1], [gc_p3,gc_c4,1], [gc_p2,gc_c2,1]],

[[gc_p2,gc_c2,1], [gc_p4,gc_c6,1], [p3,c3,1]],

[[gc_p2,gc_c2,1], [gc_p4,gc_c5,1], [p4,c4,1]]];

FullFiringSequence = [[f1-1], [gc-checking_weather_f1-1],

[gc-checking_taxiway_f1-1], [f2-1]];

AbstractedFiringSequence = [[f1-1], [fs1-1], [f2-1]];

Outcome = [['Waited before requesting permission to taxi'],

['Ground Control determined that permission should be granted for taxi'],

['Taxied to runway']];

Probability = 1.0;

Time = 25.0;

Markings from the generated marking sequence could be used with the *situation* procedure discussed earlier to give descriptions of each of the stages in the marking sequence. As in the examples of Section 6.1, the abstracted firing sequence and the corresponding descriptive outcome, is abstracted to the level indicated by the target net, in this case the Pilot net. Changing the target net has an identical effect to that illustrated in Section 6.1.

6.3 Explanation of actions

An important requirement of the EPN analysis environment is the ability to select a particular action (or group of actions) and ask the question "Why did/would this action occur?". In the case of a role-based EPN, actions will either be associated with the firing of regular transition modes or describe a surface mode. An explanation of why an action occurred or would occur is given, then, by the input requirements of the associated mode or a description of the mode summarised by a surface mode. The *explain* procedure provides this capability. For example, the goal:

explain(f2-1, RawExplanation, DescriptiveExplanation).

requires an explanation for the occurrence of mode f2-1 firing. The response to this goal would be:

RawExplanation = [[p3,c3,1]];

DescriptiveExplanation = ['Permission granted for taxi'].

That is, since the mode needing explaining is a regular transition mode, the explanation is given in the form of the input requirements for that mode. The goal:

explain(fs1-1, RawExplanation, DescriptiveExplanation).

requires an explanation for the occurrence of surface mode fs1-1 firing. The response to this goal would be:

RawExplanation = gc_fs2-1;

DescriptiveExplanation ='gc: Determined that taxiway is available for taxi'.

Requesting an explanation for this mode, ie through the goal:

explain(gc_fs2-1, RawExplanation, DescriptiveExplanation).

would give:

RawExplanation = gc-checking_taxiway_f1-1;

DescriptiveExplanation = 'gc-checking_taxiway: Determined that resource

is available'.

6.4 Sensitivity of actions that could arise from a given situation

Consider taking a specified situation and requesting an indication of the sensitivity of any actions (ie mode firings) to this situation. Sensitivity to a given situation can be determined by looking at what regular modes use the place/token combinations of the marking components that describe the situation, in their input requirements and inspect what other needs they might have before they would fire. It was decided that this capability should return all modes that are sensitive to the given situation but, whenever possible, abstract, through the use of surface modes, as high as possible up to a given target net.

The following are examples that illustrate this capability and show the changes in the descriptions as the target net is lowered. The goal:

action_sensitivity(pilot, [[p2,c2,1],[p3,c3,1]], RawOutput, DescriptiveOutput).

would return:

RawOutput = [[f2-1, [[p3,c3,1]]],

[gc_fs1-1, [[p2,c2,1],[gc_p2,gc_c2,1]]],

[fs1-2, [[p2,c2,1],[gc_p2,gc_c3,1]]]];

DescriptiveOutput = [['Taxied to runway', ['Permission granted for taxi']],

['gc: Determined that weather is OK for taxi', ['Permission to taxi requested',

'gc: Weather suitable for taxi']],

['Ground Control determined that permission should not be granted for taxi',

['Permission to taxi requested', 'gc: Weather not suitable for taxi']]].

which shows a combination of regular transition and surface modes being sensitive to the situation [[p2,c2,1], [p3,c3,1]] (ie ['Permission to taxi requested', 'Permission granted for taxi']. Also shown is the inability of gc_fs1-2 to get above the Ground Control level. Each element in the description list contains a description of the associated mode followed by a list describing the full situation required for that outcome to occur. The input requirements for each sensitive mode could be compared to the specified situation (either manually or automatically) to inspect how that situation would need to change (if at all) such that that mode would fire.

The same goal but with the target net set to Ground Control, ie:

action_sensitivity(gc, [[p2,c2,1],[p3,c3,1]], RawOutput, DescriptiveOutput).

would return:

RawOutput = [[f2-1, [[p3,c3,1]]],

[gc_fs1-1, [[p2,c2,1],[gc_p2,gc_c2,1]]],

[gc_fs1-2, [[p2,c2,1],[gc_p2,gc_c3,1]]]];

DescriptiveOutput = [['Taxied to runway', ['Permission granted for taxi']],

['gc: Determined that weather is OK for taxi', ['Permission to taxi requested',

'gc: Weather suitable for taxi']],

['gc: Determined that weather is not OK for taxi', ['Permission to taxi requested',

'gc: Weather not suitable for taxi']]].

which illustrates the effect on both forms of output of lowering the level of the target net. Lowering the level further to give the goal:

action_sensitivity(gc-checking_weather, [[p2,c2,1],[p3,c3,1]], RawOutput,

DescriptiveOutput).

would return:

RawOutput = [[f2-1, [[p3,c3,1]]],

[gc-checking_weather_f1-1, [[p2,c2,1],[gc_p2,gc_c2,1]]],

[gc-checking_weather_f1-2, [[p2,c2,1],[gc_p2,gc_c3,1]]]];

DescriptiveOutput = [['Taxied to runway', ['Permission granted for taxi']],

['gc-checking_weather: Determined that conditions are suitable',

['Permission to taxi requested', 'gc: Weather suitable for taxi']],

['gc-checking_weather: Determined that conditions are not suitable',

['Permission to taxi requested', 'gc: Weather not suitable for taxi']]].

which illustrates how the mode descriptions become less specific and more general as the target net is lowered to encapsulate a general-purpose net that has been instantiated as a role.

It is important to note that if the target net does not lie within the role hierarchy associated with a certain mode then the mode description will be given at the highest level possible. For example, the goal:

action_sensitivity(gc-checking_taxiway, [[p2,c2,1],[p3,c3,1]], RawOutput,

DescriptiveOutput).

would give the same output as that produced by the earlier example where the target net was set at the Pilot level. This eventuates since modes associated with the Checking Taxiway net are not directly sensitive to the given situation.

6.5 Actions that could result in a given situation

In order for a given situation to result, mode firings must contribute in a manner such that their combined firings meet exactly the requirements of the given situation. A procedure to provide this capability can be obtained by re-working the logic used to resolve conflict among enabled modes such that it is orientated towards mode output rather than input. That is, it determines what groups of modes can fire and result exactly in a given situation. It is important to note that firing time considerations are not made here; the procedure will return modes whose outputs upon firing cause a given situation, the modes do not necessarily fire simultaneously. Hence, a combination of timed and immediate modes could feature as the set of actions. When describing actions that could result in a situation, it was decided to provide similar features to that of Section 6.4, ie return all modes that could cause the given situation but, whenever possible, abstract, through the use of surface modes, as high as possible up to the target net.

For example, the goal:

cause_of_situation(pilot, [[p1,c1,1]], RawOutput, DescriptiveOutput).

would return:

RawOutput = [[fs1-3, [[p1,c1,1], [gc_p4,gc_c6,1]]]];

DescriptiveOutput = [['Ground Control determined that permission should not

be granted for taxi',

['Taxi not allowed', 'gc: Taxiway not available']]],

or

RawOutput = [[fs1-2, [[p1,c1,1], [gc_p2,gc_c3,1]]]];

DescriptiveOutput = [['Ground Control determined that permission should not

be granted for taxi',

['Taxi not allowed', 'gc: Weather not suitable for taxi']]],

which show that two actions could result in the situation [[p1,c1,1]] (ie ['Taxi not allowed']). Each element in the descriptive output list contains a description of the associated mode outcome followed by a list describing the full situation that that outcome would create. Similar to that mentioned in Section 6.4, the output characteristics for each mode can be compared to the specified situation (either manually or automatically) to inspect how each mode has contributed.

Changing the target net will have a similar effect as that shown in Section 6.4.

(The goal: *cause_of_situation*(pilot, [[p1,c1,2]], Raw, Desc) would result in both the above solutions being brought together signifying the need to fire both in order to create the situation required. In such cases, further investigation is necessary to determine whether such a solution is possible or whether an individual mode could fire twice and have the same effect.)

6.6 Actions that could contribute to a given situation

To determine what actions could contribute to a given situation, it is necessary to inspect mode firings whose output is sensitive to the situation concerned. That is, the result of any firing(s) might not be exactly the required situation but could contribute to it. This approach is particularly important when multiple or repeated firings of a mode is required in order to produce a given situation. This is clearly a variation on the capability presented in Section 6.4.

For the purposes of providing an example of this capability, consider the marking [[p1,c1,3]] (ie the situation ['Taxi not allowed'+3]) and requesting a description of the possible actions that could contribute to this situation. It should be noted that the request for possible contributors means that no particular marking is assumed, except that actions must contribute to components of the stated situation. The procedure that addresses this capability is *contribute_to_situation* and the goal for this example is:

contribute_to_situation(pilot, [[p1,c1,3]], RawOutput, DescriptiveOutput).

which gives:

RawOutput = [[fs1-2, [[p1,c1,1], [gc_p2,gc_c3,1]]],

[fs1-3, [[p1,c1,1], [gc_p4,gc_c6,1]]]];

DescriptiveOutput = [['Ground Control determined that permission should not

be granted for taxi',

['Taxi not allowed', 'gc: Weather not suitable for taxi']],

['Ground Control determined that permission should not be granted for taxi',

['Taxi not allowed', 'gc: Taxiway not available']]].

The output has identical features to that discussed in Section 6.5 and an identical information abstraction policy has been assumed. Inspection of the output of each mode indicates that multiple firings would be required in order to achieve the desired situation. As mentioned in Section 6.5, further investigation is necessary to determine whether this could occur.

6.7 Other issues

The labelling convention for transition mode descriptions assumes a past tense, eg "Taxied to runway" rather than "Taxiing to runway", since they represent some action that has occurred, whilst those of tokens at places assume present tense since they represent a certain state. This appears to be the best approach in the applications considered to date. Note also that places are not given labels, only the tokens that can reside at a place are described. In general, transition mode descriptions indicate what happened whilst place/token descriptions indicate why.

Consider briefly the example shown in Figure 11, with associated Prolog database representation in Figure 12. Consider generating the sequence of actions stemming from the situation ['Taxi not allowed', 'Weather suitable for traffic', 'Taxiway available', 'Airway available'] and resulting in some situation that includes ['Airborne']. The associated goal is:

sequence_of_actions(pilot, [[p1,c1,1], [p5,c5,1], [p6,c7,1], [p10,c12,1]],

[[p9,c11,1] | _], _, FullFiringSequence,

AbstractedFiringSequence, Outcome, Probability, Time, _).

giving:

FullFiringSequence = [[f1-1], [gc-checking_conditions_f1-1],

[gc-checking_resource_f1-1], [f2-1],

[ac-checking_conditions_f1-1], [ac-checking_resource_f1-1], [f4-1]];

AbstractedFiringSequence = [[f1-1], [fs1-1], [f2-1], [fs2-1], [f4-1]];

Outcome = [['Waited before requesting permission to taxi'],

['Ground Control determined that permission should be granted for taxi'],

['Taxied to runway'],

['Air Control determined that permission should be granted for take off'],

['Took off']];

Probability = 1.0;

Time = 45.0 .

which is presented in a clear descriptive form. However, requiring an explanation for why mode fs1-1 fired (ie using the *explain* procedure discussed in Section 6.3) would give:

RawExplanation = gc_fs2-1;

DescriptiveExplanation ='gc: Determined that resource is available'.

which is a very general explanation and leaves the enquirer asking "What resource?". Although this could be determined through the repeated use of the *explain* procedure, starting on mode gc_fs2-1, this illustrates that the explanation capability is only as good as the quality of the descriptions tagged to modes and place/token combinations. In the case of Figure 11, a general controller was instantiated as both a Ground Control and an Air Control net. The original descriptions for the Controller net were maintained and only tagged automatically by the role name at instantiation. This illustrates that it will be preferable, on occasions, to manually modify the descriptions when generic roles are employed in order to create more meaningful explanations.

7. Discussion

In the case of the DICE simulation, adequate credibility and realism in EPN artificial agents is essential and judgement of these qualities can generally be best made by military domain experts. Such an expert should be able to interrogate features of the agents and to form some judgement on the realism of that artificial agent compared with the real-world system that the artificial agent represents. Having military endorsement of the assumptions used in such representations is a vital prerequisite to any C3I system study.

The explanation and analysis tool reported in this document can be regarded as the provider of information that facilitates achievement of military endorsement. However, the power of such a tool can be somewhat lost if that information is not presented in a some user-friendly and clear manner that also shelters the domain expert from the terminology and notation perculiar to Petri nets. For example, the *situation* procedure could be used to convey descriptions of possible situations that the user could select from when defining a situation or state to the analysis environment, an alternative to graphically placing tokens within a Petri net diagram.

The explanation capability is only as good as the quality of the information tagged to the EPN. Also, the process of tailoring role features upon instantiation is considered important but may become a laborious task in practice if role nets are large and

complex. Furthermore, an assumption has been made that the abstraction level of a user can be translated into a corresponding target net. All of the above issues and the general applicability and limitations of the current tool will be addressed as the tool receives greater exposure through application.

8. Conclusions

An explanation and analysis capability has been developed for a class of extended Petri nets (EPN). This capability provides a stand-alone analysis tool for use with any EPN. In the case of C3I applications, that EPN might represent a C3I system, sub-system or node where analysis could include inspection of connectivity, information flow and response times. Furthermore, the explanation and analysis tool can be used to convey the underlying characteristics of the EPN to a domain expert, knowledgeable in the real or proposed system that that EPN represents. Integrated within a suitable graphical user interface environment, such a capability facilitates quick revision of the EPN and subsequent re-analysis and convergence on a satisfactory representation for the task at hand.

Research is ongoing into addressing further the transient and stochastic features that an artificial agent can possess. Enhancements to the current capability will be made as this research bears fruit.

9. References

1. Davies, M. and Gabrisch, C. (1995), "The Distributed Interactive C3I Effectiveness (DICE) simulation project: an overview", Proc. *5th Conf. Computer Generated Forces and Behavioral Representation*, Orlando, Florida, May 9-11, Pub. by Institute for Simulation and Training, Florida.

2. Johnson, W.L. (1994), "Agents that Explain their Own Actions", Proc. *4th Conf. Computer Generated Forces and Behavioral Representation*, Orlando, Florida, May 4-6, Pub. by Institute for Simulation and Training, Florida.

3. Bowden, F.D.J., Davies, M., and Dunn, J.M. (1995), "Representing Role-based Agents using Coloured Petri Nets", Proc. *5th Conf. Computer Generated Forces and Behavioral Representation*, Orlando, Florida, May 9-11, Pub. by Institute for Simulation and Training, Florida.

4. Bratko, I. (1990), *"Prolog programming for artificial intelligence"*, 2nd ed., Addison-Wesley Pub. Ltd, London.

5. Quintus Corporation (1991), *"Quintus Prolog, Release 3. User Manuals"*, California.

6. Cox, A., Gibb, A. and I. Page (1994), "Army Training and CGFs - A UK Perspective", Proc. *4th Conf. Computer Generated Forces and Behavioral Representation*, Orlando, Florida, May 4-6, Pub. by Institute for Simulation and Training, Florida.

7. Bowden, F.D.J. (1996), "Modelling time in Petri nets", Proc. *2nd Australia-Japan Workshop on Stochastic Models*, Gold Coast, Australia

8. Ajmone Marsan, M., Baldo, G., and Conte, G. (1984), "A Class of Generalized Stochastic Petri Nets for the Performance Analysis of Multiprocessor Systems", ACM Transaction on Computer Systems, 2(1), May

9. Dunn, J.M., Davies, M. (1996), "Extended Petri Net Explanation and Analysis Capability - Software Specification", DSTO General Document DSTO-GD-0104, September

This page intentionally left blank

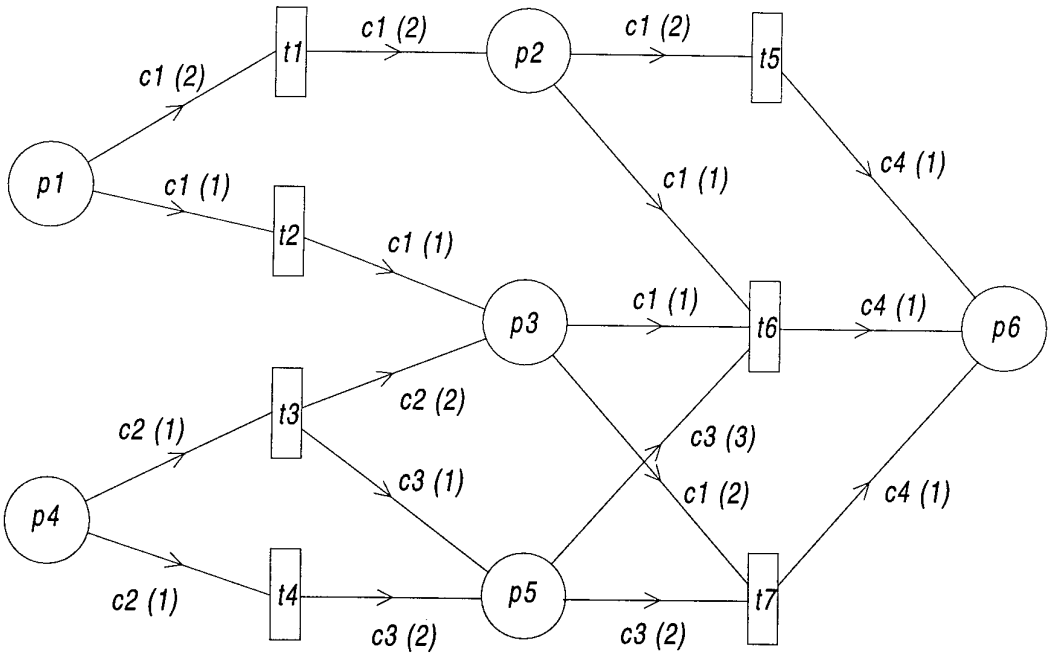
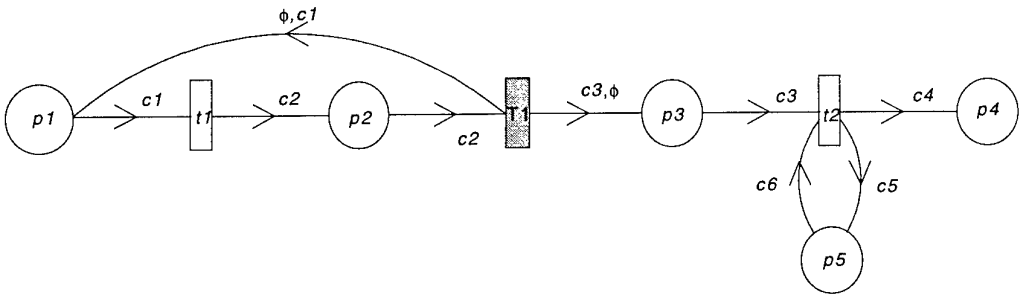
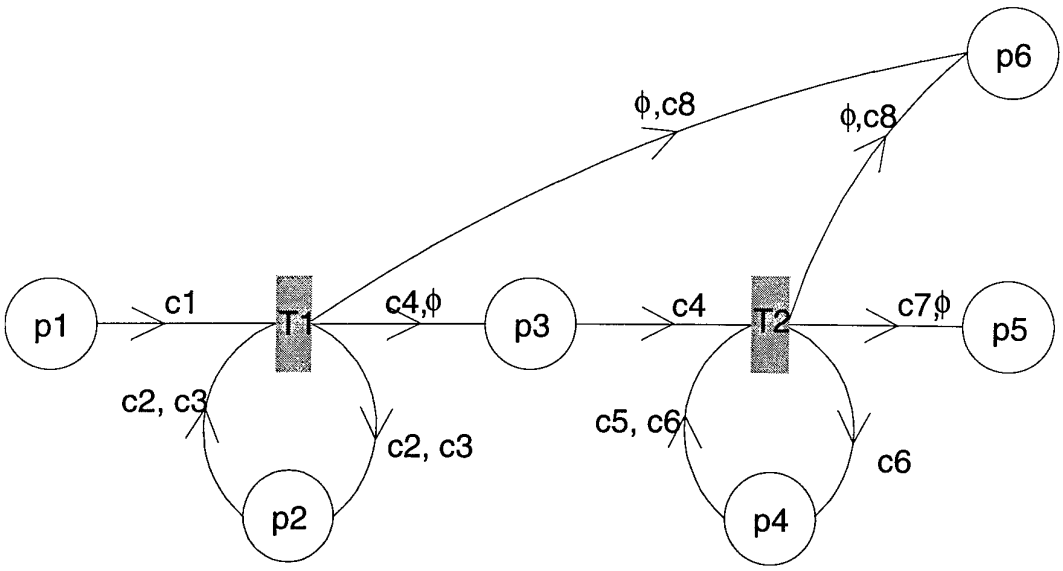


Figure 1: Sample EPN



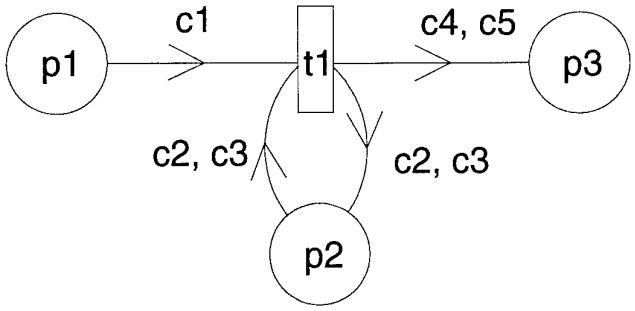
Place/Token Combination	Description
p1, c1	Taxi not allowed
p2, c2	Permission to taxi requested
p3, c3	Permission granted for taxi
p4, c4	At runway
p5, c5	Taxiway available
p5, c6	Taxiway unavailable

Figure 2: EPN representing Pilot activities



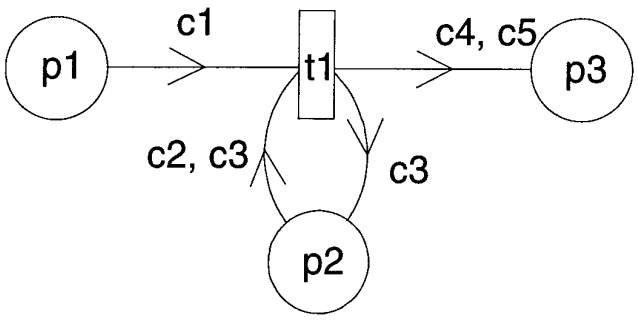
Place/Token Combination	Description
p1, c1	Permission to taxi requested
p2, c2	Weather suitable for taxi
p2, c3	Weather not suitable for taxi
p3, c4	Weather allows taxi
p4, c5	Taxiway available
p4, c6	Taxiway not available
p5, c7	Permission granted for taxi
p6, c8	Permission not granted for taxi

Figure 3: EPN representing Ground Control activities



Place/Token Combination	Description
p1, c1	Request to check conditions received
p2, c2	Conditions OK
p2, c3	Conditions not OK
p3, c4	Conditions OK
p3, c5	Conditions not OK

Figure 4: EPN to check conditions



Place/Token Combination	Description
p1, c1	Request to check availability of resource received
p2, c2	Resource available
p2, c3	Resource not available
p3, c4	Resource available
p3, c5	Resource not available

Figure 5: EPN to check resource availability

```
/* DATABASE */

/* modes/1 */
/* List(1) of transition modes applicable to this Petri net */
modes([f1-1,f2-1,f3-1,f4-1,f5-1,f6-1,f7-1]).

/* trans_modes/2 */
/* Assigns modes(2) to a transition(1). */
trans_modes(t1, [f1-1]).
trans_modes(t2, [f2-1]).
trans_modes(t3, [f3-1]).
trans_modes(t4, [f4-1]).
trans_modes(t5, [f5-1]).
trans_modes(t6, [f6-1]).
trans_modes(t7, [f7-1]).

/* trans_mode/5 */
/* Specifies transition mode(2) descriptions(3) and associated net(1). Parameters (4) and (5)
  indicate mode weight/measure-of-dispersion and mean firing time respectively. */
trans_mode(petri_net, f1-1, 'Mode f1-1 - a timed mode', 0, 3.2).
trans_mode(petri_net, f2-1, 'Mode f2-1 - a timed mode', 0, 5.0).
trans_mode(petri_net, f3-1, 'Mode f3-1 - an immediate mode', 2, 0).
trans_mode(petri_net, f4-1, 'Mode f4-1 - a timed mode', 0, 8.2).
trans_mode(petri_net, f5-1, 'Mode f5-1 - an immediate_mode', 4, 0).
trans_mode(petri_net, f6-1, 'Mode f6-1 - an immediate mode', 1, 0).
trans_mode(petri_net, f7-1, 'Mode f7-1 - an immediate mode', 1, 0).

/* trans_mode_input/2 */
/* Specified input requirements(2) for a mode(1). */
trans_mode_input(f1-1, [[p1,c1,2]]).
trans_mode_input(f2-1, [[p1,c1,1]]).
trans_mode_input(f3-1, [[p4,c2,1]]).
trans_mode_input(f4-1, [[p4,c2,1]]).
trans_mode_input(f5-1, [[p2,c1,2]]).
trans_mode_input(f6-1, [[p2,c1,1],
                        [p3,c1,1],
                        [p5,c3,3]]).
trans_mode_input(f7-1, [[p3,c1,2],
                        [p5,c3,2]]).

/* trans_mode_output/2 */
/* Describes output characteristics(2) for a mode(1). */
trans_mode_output(f1-1, [[p2,c1,2]]).
trans_mode_output(f2-1, [[p3,c1,1]]).
trans_mode_output(f3-1, [[p3,c2,2],
                         [p5,c3,1]]).
trans_mode_output(f4-1, [[p5,c3,2]]).
trans_mode_output(f5-1, [[p6,c4,1]]).
trans_mode_output(f6-1, [[p6,c4,1]]).
trans_mode_output(f7-1, [[p6,c4,1]]).
```

Figure 6: Prolog representation of sample EPN

```
/*   DATABASE   */

/* modes/1 */
/* List(1) of transition modes applicable to this Petri net */
modes([f1-1, f1-2]).

/* trans_mode/5 */
/* Specifies transition mode(2) descriptions(3) and associated net(1). Parameters (4) and (5)
   indicate mode weight/measure-of-dispersion and mean firing time respectively. */
trans_mode(check_conditions, f1-1, 'Determined that conditions are suitable', 0, 6.0).
trans_mode(check_conditions, f1-2, 'Determined that conditions are not suitable', 0, 6.0).

/* trans_mode_input/2 */
/* Specified input requirements(2) for a transition mode(1). */
trans_mode_input(f1-1, [[p1,c1,1], [p2,c2,1]]).
trans_mode_input(f1-2, [[p1,c1,1], [p2,c3,1]]).

/* trans_mode_output/2 */
/* Describes output characteristics(2) for a transition mode(1). */
trans_mode_output(f1-1, [[p2,c2,1], [p3,c4,1]]).
trans_mode_output(f1-2, [[p2,c3,1], [p3,c5,1]]).

/* place_token/3 */
/* Gives a description(3) for a given place(1) and token(2) */
place_token(p1, c1, 'Request to check conditions received').
place_token(p2, c2, 'Conditions OK').
place_token(p2, c3, 'Conditions not OK').
place_token(p3, c4, 'Conditions OK').
place_token(p3, c5, 'Conditions not OK').
```

Figure 7: Prolog representation of Check Conditions EPN

```
/*   DATABASE   */

/* modes/1 */
/* List(1) of transition modes applicable to this Petri net */
modes([f1-1, f1-2]).

/* trans_mode/5 */
/* Specifies transition mode(2) descriptions(3) and associated net(1). Parameters (4) and (5)
  indicate mode weight/measure-of-dispersion and mean firing time respectively. */
trans_mode(check_availability_of_resource, f1-1, 'Determined that resource is available', 0, 4.0).
trans_mode(check_availability_of_resource, f1-2, 'Determined that resource is not available', 0, 4.0).

/* trans_mode_input/2 */
/* Specified input requirements(2) for a transition mode(1). */
trans_mode_input(f1-1, [[p1,c1,1], [p2,c2,1]]).
trans_mode_input(f1-2, [[p1,c1,1], [p2,c3,1]]).

/* trans_mode_output/2 */
/* Describes output characteristics(2) for a transition mode(1). */
trans_mode_output(f1-1, [[p2,c3,1], [p3,c4,1]]).
trans_mode_output(f1-2, [[p2,c3,1], [p3,c5,1]]).

/* place_token/3 */
/* Gives a description(3) for a given place(1) and token(2) */
place_token(p1, c1, 'Request to check the availability of resource received').
place_token(p2, c2, 'Resource available').
place_token(p2, c3, 'Resource not available').
place_token(p3, c4, 'Resource available').
place_token(p3, c5, 'Resource not available').
```

Figure 8: Prolog representation of Check Availability of Resource EPN

```
/*   DATABASE   */

/* modes/1 */
/* List(1) of transition modes applicable to this Petri net */
modes([checking_weather_f1-1, checking_weather_f1-2,
   checking_taxiway_f1-1, checking_taxiway_f1-2]).

/* trans_mode/5 */
/* Specifies transition mode(2) descriptions(3) and associated net(1). Parameters (4) and (5)
  indicate mode weight/measure-of-dispersion and mean firing time respectively. */
trans_mode(checking_weather, checking_weather_f1-1,
        'checking_weather: Determined that conditions are suitable', 0, 6.0).
trans_mode(checking_weather, checking_weather_f1-2,
        'checking_weather: Determined that conditions are not suitable', 0, 6.0).
trans_mode(checking_taxiway, checking_taxiway_f1-1,
        'checking_taxiway: Determined that resource is available', 0, 4.0).
trans_mode(checking_taxiway, checking_taxiway_f1-2,
        'checking_taxiway: Determined that resource is not available', 0, 4.0).

/* trans_mode_input/2 */
/* Specified input requirements(2) for a transition mode(1). */
trans_mode_input(checking_weather_f1-1, [[p1,c1,1], [p2,c2,1]]).
trans_mode_input(checking_weather_f1-2, [[p1,c1,1], [p2,c3,1]]).
trans_mode_input(checking_taxiway_f1-1, [[p3,c4,1], [p4,c5,1]]).
trans_mode_input(checking_taxiway_f1-2, [[p3,c4,1], [p4,c6,1]]).

/* trans_mode_output/2 */
/* Describes output characteristics(2) for a transition mode(1). */
trans_mode_output(checking_weather_f1-1, [[p2,c2,1], [p3,c4,1]]).
trans_mode_output(checking_weather_f1-2, [[p6,c8,1], [p2,c3,1]]).
trans_mode_output(checking_taxiway_f1-1, [[p5,c7,1], [p4,c6,1]]).
trans_mode_output(checking_taxiway_f1-2, [[p6,c8,1], [p4,c6,1]]).

/* place_token/3 */
/* Gives a description(3) for a given place(1) and token(2) */
place_token(p1, c1, 'Permission to taxi requested').
place_token(p2, c2, 'Weather suitable for taxi').
place_token(p2, c3, 'Weather not suitable for taxi').
place_token(p3, c4, 'Weather allows taxi').
place_token(p4, c5, Taxiway available').
place_token(p4, c6, Taxiway not available').
place_token(p5, c7, 'Permission granted for taxi').
place_token(p5, c8, 'Permission not granted for taxi').

/* surface_mode/5 */
/* Takes an output possibility from a role and describes it.
   An output possibility is any firing that results in external distribution of tokens from the role net.
   Describes a role surface mode(2), its internal equivalent firing(3) and summary description(4).
   Also given is the net(1) that uses the role and the role name(5). */
surface_mode(gc, fs1-1, checking_weather_f1-1,
        'Determined that weather is OK for taxi', checking_weather).
surface_mode(gc, fs1-2, checking_weather_f1-2,
        'Determined that weather is not OK for taxi',checking_weather).
surface_mode(gc, fs2-1, checking_taxiway_f1-1,
        'Determined that taxiway is available for taxi', checking_taxiway).
surface_mode(gc, fs2-2, checking_taxiway_f1-2,
        'Determined that taxiway is not available for taxi', checking_taxiway).
```

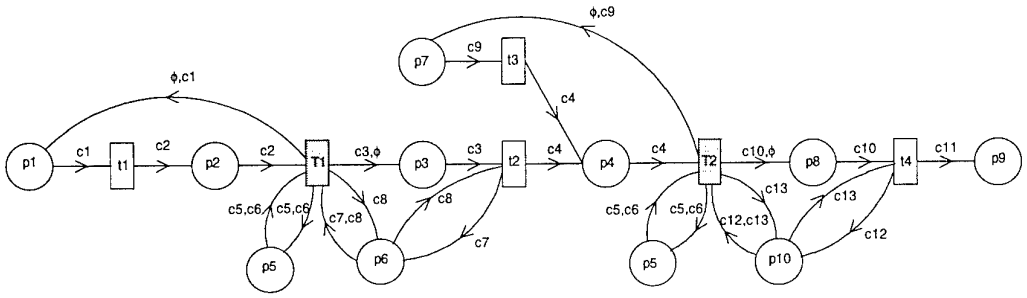
Figure 9: Prolog representation of Ground Control EPN

```
/*    DATABASE    */

/* modes/1 */
/* List(1) of transition modes applicable to this Petri net */
modes([f1-1, f2-1, gc-checking_weather_f1-1, gc-checking_weather_f1-2,
  gc-checking_taxiway_f1-1, gc-checking_taxiway_f1-2]).

/* trans_mode/5 */
/* Specifies transition mode(2) descriptions(3) and associated net(1). Parameters (4) and (5)
  indicate mode weight/measure-of-dispersion and mean firing time respectively. */
trans_mode(pilot, f1-1, 'Waited before requesting permission to taxi', 0, 5.0).
trans_mode(pilot, f2-1, 'Taxied to runway', 0, 10.0).
trans_mode(gc-checking_weather, gc-checking_weather_f1-1,
        'gc-checking_weather: Determined that conditions are suitable', 0, 6.0).
trans_mode(gc-checking_weather, gc-checking_weather_f1-2,
        'gc-checking_weather: Determined that conditions are not suitable', 0, 6.0).
trans_mode(gc-checking_taxiway, gc-checking_taxiway_f1-1,
        'gc-checking_taxiway: Determined that resource is available', 0, 4.0).
trans_mode(gc-checking_taxiway, gc-checking_taxiway_f1-2,
        'gc-checking_taxiway: Determined that resource is not available', 0, 4.0).

/* trans_mode_input/2 */
/* Specified input requirements(2) for a transition mode(1). */
trans_mode_input(f1-1, [[p1,c1,1]]).
trans_mode_input(f2-1, [[p3,c3,1], [gc_p4,gc_c6,1]]).
trans_mode_input(gc-checking_weather_f1-1, [[p2,c2,1], [gc_p2,gc_c2,1]]).
trans_mode_input(gc-checking_weather_f1-2, [[p2,c2,1], [gc_p2,gc_c3,1]]).
trans_mode_input(gc-checking_taxiway_f1-1, [[gc_p3,gc_c4,1], [gc_p4,gc_c5,1]]).
trans_mode_input(gc-checking_taxiway_f1-2, [[gc_p3,gc_c4,1], [gc_p4,gc_c6,1]]).

/* trans_mode_output/2 */
/* Describes output characteristics(2) for a transition mode(1). */
trans_mode_output(f1-1, [[p2,c2,1]]).
trans_mode_output(f2-1, [[p4,c4,1], [gc_p4,gc_c5,1]]).
trans_mode_output(gc-checking_weather_f1-1, [[gc_p2,gc_c2,1], [gc_p3,gc_c4,1]]).
trans_mode_output(gc-checking_weather_f1-2, [[p1,c1,1], [gc_p2,gc_c3,1]]).
trans_mode_output(gc-checking_taxiway_f1-1, [[p3,c3,1], [gc_p4,gc_c6,1]]).
trans_mode_output(gc-checking_taxiway_f1-2, [[p1,c1,1], [gc_p4,gc_c6,1]]).

/* place_token/3 */
/* Gives a description(3) for a given place(1) and token(2) */
place_token(p1, c1, 'Taxi not allowed').
place_token(p2, c2, 'Permission to taxi requested').
place_token(p3, c3, 'Permission granted for taxi').
place_token(p4, c4, 'At runway').
place_token(gc_p2, gc_c2, 'gc: Weather suitable for taxi').
place_token(gc_p2, gc_c3, 'gc: Weather not suitable for taxi').
place_token(gc_p3, gc_c4, 'gc: Weather allows taxi').
place_token(gc_p4, gc_c5, 'gc: Taxiway available').
place_token(gc_p4, gc_c6, 'gc: Taxiway not available').
```

Figure 10: Prolog representation of Pilot EPN

```
/* surface_mode/5 */
/* Takes an output possibility from a role and describes it.
   An output possibility is any firing that results in external distribution of tokensfrom the role net.
   Describes a role surface mode(2), its internal equivalent firing(3) and summary description(4).
   Also given is the net(1) that uses the role and the role name(5). */
surface_mode(pilot, fs1-1, gc_fs2-1,
        'Ground Control determined that permission should be granted for taxi', gc).
surface_mode(pilot, fs1-2, gc_fs1-2,
        'Ground Control determined that permission should not be granted for taxi', gc).
surface_mode(pilot, fs1-3, gc_fs2-2,
        'Ground Control determined that permission should not be granted for taxi', gc).
surface_mode(gc, gc_fs1-1, gc-checking_weather_f1-1,
        'gc: Determined that weather is OK for taxi', gc-checking_weather).
surface_mode(gc, gc_fs1-2, gc-checking_weather_f1-2,
        'gc: Determined that weather is not OK for taxi',gc-checking_weather).
surface_mode(gc, gc_fs2-1, gc-checking_taxiway_f1-1,
        'gc: Determined that taxiway is available for taxi', gc-checking_taxiway).
surface_mode(gc, gc_fs2-2, gc-checking_taxiway_f1-2,
        'gc: Determined that taxiway is not available for taxi', gc-checking_taxiway).
```

Figure 10: Prolog representation of Pilot EPN (cont'd)



Place/Token Combination	Description
p1, c1	Taxi not allowed
p2, c2	Permission to taxi requested
p3, c3	Permission granted for taxi
p4, c4	At runway, permission to take off requested
p5, c5	Weather suitable for traffic
p5, c6	Weather not suitable for traffic
p6, c7	Taxiway available
p6, c8	Taxiway not available
p7, c9	Take off not allowed
p8, c10	Permission granted for take off
p9, c11	Airborne
p10, c12	Airway available
p10, c13	Airway not available

Figure 11: EPN representing Pilot activities incorporating taxiing and take-off

```
/*   DATABASE   */
/* modes/1 */
/* List(1) of transition modes applicable to this Petri net */
modes([f1-1, f2-1, f3-1, f4-1, gc-checking_conditions_f1-1, gc-checking_conditions_f1-2,
  gc-checking_resource_f1-1, gc-checking_resource_f1-2, ac-checking_conditions_f1-1,
  ac-checking_conditions_f1-2, ac-checking_resource_f1-1, ac-checking_resource_f1-2]).

/* trans_mode/5 */
/* Specifies transition mode(2) descriptions(3) and associated net(1). Parameters (4) and (5)
  indicate mode weight/measure-of-dispersion and mean firing time respectively. */
trans_mode(pilot, f1-1, 'Waited before requesting permission to taxi', 0, 5.0).
trans_mode(pilot, f2-1, 'Taxied to runway', 0, 10.0).
trans_mode(pilot, f3-1, 'Waited before requesting permission to take off', 0, 5.0).
trans_mode(pilot, f4-1, 'Took off', 0, 10.0).
trans_mode(gc-checking_conditions, gc-checking_conditions_f1-1,
        'gc-checking_conditions: Determined that conditions are suitable', 0, 6.0).
trans_mode(gc-checking_conditions, gc-checking_conditions_f1-2,
        'gc-checking_conditions: Determined that conditions are not suitable', 0, 6.0).
trans_mode(gc-checking_resource, gc-checking_resource_f1-1,
        'gc-checking_resource: Determined that resource is available', 0, 4.0).
trans_mode(gc-checking_resource, gc-checking_resource_f1-2,
        'gc-checking_resource: Determined that resource is not available', 0, 4.0).
trans_mode(ac-checking_conditions, ac-checking_conditions_f1-1,
        'ac-checking_conditions: Determined that conditions are suitable', 0, 6.0).
trans_mode(ac-checking_conditions, ac-checking_conditions_f1-2,
        'ac-checking_conditions: Determined that conditions are not suitable', 0, 6.0).
trans_mode(ac-checking_resource, ac-checking_resource_f1-1,
        'ac-checking_resource: Determined that resource is available', 0, 4.0).
trans_mode(ac-checking_resource, ac-checking_resource_f1-2,
        'ac-checking_resource: Determined that resource is not available', 0, 4.0).

/* trans_mode_input/2 */
/* Specified input requirements(2) for a transition mode(1). */
trans_mode_input(f1-1, [[p1,c1,1]]).
trans_mode_input(f2-1, [[p3,c3,1]]).
trans_mode_input(f3-1, [[p7,c9,1]]).
trans_mode_input(f4-1, [[p8,c10,1]]).
trans_mode_input(gc-checking_conditions_f1-1, [[p2,c2,1], [p5,c5,1]]).
trans_mode_input(gc-checking_conditions_f1-2, [[p2,c2,1], [p5,c6,1]]).
trans_mode_input(gc-checking_resource_f1-1, [[gc_p3,gc_c4,1], [p6,c7,1]]).
trans_mode_input(gc-checking_resource_f1-2, [[gc_p3,gc_c4,1], [p6,c8,1]]).
trans_mode_input(ac-checking_conditions_f1-1, [[p4,c4,1], [p5,c5,1]]).
trans_mode_input(ac-checking_conditions_f1-2, [[p4,c4,1], [p5,c6,1]]).
trans_mode_input(ac-checking_resource_f1-1, [[ac_p3,ac_c4,1], [p10,c12,1]]).
trans_mode_input(ac-checking_resource_f1-2, [[ac_p3,ac_c4,1], [p10,c13,1]]).

/* trans_mode_output/2 */
/* Describes output characteristics(2) for a transition mode(1). */
trans_mode_output(f1-1, [[p2,c2,1]]).
trans_mode_output(f2-1, [[p4,c4,1], [p6,c7,1]]).
trans_mode_output(f3-1, [[p4,c4,1]]).
trans_mode_output(f4-1, [[p9,c11,1], [p10,c12,1]]).
trans_mode_output(gc-checking_conditions_f1-1, [[p5,c5,1], [gc_p3,gc_c4,1]]).
trans_mode_output(gc-checking_conditions_f1-2, [[p1,c1,1], [p5,c6,1]]).
trans_mode_output(gc-checking_resource_f1-1, [[p3,c3,1], [p6,c8,1]]).
trans_mode_output(gc-checking_resource_f1-2, [[p1,c1,1], [p6,c8,1]]).
trans_mode_output(ac-checking_conditions_f1-1, [[p5,c5,1], [ac_p3,ac_c4,1]]).
trans_mode_output(ac-checking_conditions_f1-2, [[p5,c6,1], [p7,c9,1]]).
trans_mode_output(ac-checking_resource_f1-1, [[p10,c13,1], [p8,c10,1]]).
trans_mode_output(ac-checking_resource_f1-2, [[p10,c13,1], [p7,c9,1]]).
```

Figure 12: Prolog representation of Pilot activities incorporating taxiing and take-off

```
/* place_token/3 */
/* Gives a description(3) for a given place(1) and token(2) */
place_token(p1, c1, 'Taxi not allowed').
place_token(p2, c2, 'Permission to taxi requested').
place_token(p3, c3, 'Permission granted for taxi').
place_token(p4, c4, 'At runway, permission to take off requested').
place_token(p5, c5, 'Weather suitable for traffic').
place_token(p5, c6, 'Weather not suitable for traffic').
place_token(p6, c7, 'Taxiway available').
place_token(p6, c8, 'Taxiway not available').
place_token(p7, c9, 'Take off not allowed').
place_token(p8, c10, 'Permission granted for take off').
place_token(p9, c11, 'Airborne').
place_token(p10, c12, 'Airway available').
place_token(p10, c13, 'Airway not available').
place_token(gc_p3, gc_c4, 'gc: Conditions suitable').
place_token(ac_p3, ac_c4, 'ac: Conditions suitable').

/* surface_mode/5 */
/* Takes an output possibility from a role and describes it.
   An output possibility is any firing that results in external distribution of tokensfrom the role net.
   Describes a role surface mode(2), its internal equivalent firing(3) and summary description(4).
   Also given is the net(1) that uses the role and the role name(5). */
surface_mode(pilot, fs1-1, gc_fs2-1,
        'Ground Control determined that permission should be granted for taxi', gc).
surface_mode(pilot, fs1-2, gc_fs1-2,
        'Ground Control determined that permission should not be granted for taxi', gc).
surface_mode(pilot, fs1-3, gc_fs2-2,
        'Ground Control determined that permission should not be granted for taxi', gc).
surface_mode(pilot, fs2-1, ac_fs2-1,
        'Air Control determined that permission should be granted for take off', ac).
surface_mode(pilot, fs2-2, ac_fs1-2,
        'Air Control determined that permission should not be granted for take off', ac).
surface_mode(pilot, fs2-3, ac_fs2-2,
        'Air Control determined that permission should not be granted for take off', ac).
surface_mode(gc, gc_fs1-1, gc-checking_conditions_f1-1,
        'gc: Determined that conditions are OK', gc-checking_conditions).
surface_mode(gc, gc_fs1-2, gc-checking_conditions_f1-2,
        'gc: Determined that conditions are not OK',gc-checking_conditions).
surface_mode(gc, gc_fs2-1, gc-checking_resource_f1-1,
        'gc: Determined that resource is available', gc-checking_resource).
surface_mode(gc, gc_fs2-2, gc-checking_resource_f1-2,
        'gc: Determined that resource is not available', gc-checking_resource).
surface_mode(ac, ac_fs1-1, ac-checking_conditions_f1-1,
        'ac: Determined that conditions are OK', ac-checking_conditions).
surface_mode(ac, ac_fs1-2, ac-checking_conditions_f1-2,
        'ac: Determined that conditions are not OK',ac-checking_conditions).
surface_mode(ac, ac_fs2-1, ac-checking_resource_f1-1,
        'ac: Determined that resource is available', ac-checking_resource).
surface_mode(ac, ac_fs2-2, ac-checking_resource_f1-2,
        'ac: Determined that resource is not available', ac-checking_resource).
```

Figure 12: Prolog representation of Pilot activities incorporating taxiing and take-off (cont'd)

Appendix A

Firing Probabilities for Immediate Transitions

A1 Introduction

Within an EPN simulation, execution strategies are employed whenever more than one transition mode is enabled. The chosen execution strategy is discussed in Section 2.2 of the main text and is employed to resolve any conflict and determine which set of mode firings will occur. The chosen set or solution is generally one of a number of possible outcomes that could arise. In the case of the Prolog EPN explanation and analysis software each outcome is generated with some associated probability.

One important feature of the EPN explanation and analysis programme, is the ability to generate and analyse such probabilities and also to inspect the transient features of an EPN. The following sections detail how such probabilities are calculated for modes of immediate transitions; probabilistic analysis for timed transitions is the subject of ongoing research.

Consider the situation where a certain marking exists which has enabled a number of immediate modes. From the list of enabled modes, sets of dependent modes are generated; these groups are then used in the selection of modes for firing. The following sections detail this process, including the implementation in Prolog and the associated firing probability computation.

A2 Determining dependent mode sets

Dependent mode sets are groups of modes that are competing for common tokens of the EPN marking. Taking each component of an EPN marking in turn, dependent mode sets are determined by the following process:

(i) From the enabled modes, determine the list of modes which demand, through their input requirements, tokens associated with that component;

(ii) Compare total demand of the list with number of tokens associated with marking component;

(a) If demand exceeds supply then:

- The list of modes forms a new dependent mode set;
- If the new set is a subset of another previously identified set then disregard the new one;
- If the new set can contain the members of some previously identified set then disregard the previous one;

- If any members of the new set are currently recommended for firing then cancel those particular recommendations.

(b) If demand is less than supply:

- Recommend list of modes for firing.

This procedure results in a list of modes that are already approved for firing and a list of dependent mode sets that require further processing.

A3 Sampling modes for firing

Multiple immediate transition modes can be fired simultaneously; the modes to be fired are determined, in an EPN simulation, through a probabilistic selection process. The firing probabilities calculated in the Prolog software must reflect this process. The basic characteristics of this process can be conveyed by considering the selection of one mode from a dependent mode set that holds three modes f1, f2 and f3 with weights 1, 2 and 3 respectively. A uniform distribution is formed which spans the sum of the weights of the modes concerned; ie for this example the distribution takes the form:

1	2	3
f1	f2	f3

A random sample over this distribution determines which mode should fire. The probability of f1 firing is, therefore, 1/(1+2+3) = 1/6; f2 firing = 2/6; and f3 firing= 3/6.

This process becomes slightly more complex when more than one dependent set of modes exists and where the sets overlap through the presence of common members. It is important to illustrate the complete process through the use of a number of examples.

A3.1 Example 1

Consider the independent mode sets, A and B given by:

(A) : [f1, f2, f3];

(B): [f4, f5]

where the respective mode weights are:

(A) : [1, 2, 3];

(B): [4, 5] .

In this example the dependent sets do not intersect and it is assumed that only one mode can fire within a given set.

The uniform distribution, for this example, is:

1	2	3	4	5
f1	f2	f3	f4	f5

Sampling over this band of width 15 works in the manner described earlier except that now when a certain mode gets picked, members of its dependent mode set are removed from the band (since, in this example, only one mode can fire within a given mode set) and are not considered in future sampling. The sampling band therefore shrinks as samples are made. For example, consider the case where f4 were picked first (with probability 4/15). The mode f4 belongs to a set with mode f5 and hence the band now becomes:

1	2	3
f1	f2	f3

Sampling continues and if mode f3 were to be picked this time then the associated probability would be 3/6.

For Example 1, there are 12 outcomes that could occur:

[f1 picked, followed by **f4]**; **[f4 picked**, followed by **f1]**

[f1 picked, followed by **f5]**; **[f5 picked**, followed by **f1]**

[f2 picked, followed by **f4]**; **[f4 picked**, followed by **f2]**

[f2 picked, followed by **f5]**; **[f5 picked**, followed by **f2]**

[f3 picked, followed by **f4]**; **[f4 picked**, followed by **f3]**

[f3 picked, followed by **f5]**; **[f5 picked**, followed by **f3]**

(It should be noted that the firings in any given outcome are carried out simultaneously, the order refers to selection of which modes to fire.) Each of these outcomes has an associated probability. Denoting such outcomes in the form [f2, f5], indicating f2 picked first followed by f5, the probabilities associated with these outcomes are:

[f1, f4] : (1/15)(4/9) = 4/135; **[f4, f1]** : (4/15)(1/6) = 2/45;

[f1, f5] : (1/15)(5/9) = 1/27; **[f5, f1]** : (5/15)(1/6) = 1/18;

[f2, f4] : (2/15)(4/9) = 8/135; **[f4, f2]** : (4/15)(2/6) = 4/45;

[f2, f5] : (2/15)(5/9) = 2/27; **[f5, f2]** : (5/15)(2/6) = 1/9;

[f3, f4] : (3/15)(4/9) = 4/45; **[f4, f3]** : (4/15)(3/6) = 2/15;

[f3, f5] : (3/15)(5/9) = 1/9; **[f5,f3]** : (5/15)(3/6) = 1/6.

As expected, the total of the above probabilities is unity which reflects the likelihood of getting any one of the possible solutions. Note also that the probability of f1 firing in any solution is the sum of the probabilities for solutions [f1, f4], [f4, f1], [f1, f5], [f5, f1] which equals 1/6 which is the same value as that obtained from considering set (A) in isolation. This characteristic is a direct consequence of the dependent mode sets not intersecting.

A3.2 Example 2

Consider the independent mode sets:

(A) : [f1, f2, f3];

(B) : [f3, f4];

(C) : [f4, f5]

with respective weights:

(A) : [1, 2, 3];

(B) : [3, 4];

(C) : [4, 5] .

In this example there are three dependent sets which do intersect but it is still assumed that only one mode can fire within a given set. The strategy adopted in Example 1 will again be used here.

In this example, there are 10 possible outcomes with the following associated probabilities:

[f1, f4] :	(1/15)(4/9) = 4/135;	**[f4, f1]** :	(4/15)(1/3) = 4/45;
[f1, f5] :	(1/15)(5/9) = 1/27;	**[f5, f1]** :	(5/15)(1/6) = 1/18;
[f2, f4] :	(2/15)(4/9) = 8/135;	**[f4, f2]** :	(4/15)(2/3) = 8/45;
[f2, f5] :	(2/15)(5/9) = 2/27;	**[f5, f2]** :	(5/15)(2/6) = 1/9;
[f3, f5] :	(3/15) = 1/5;	**[f5,f3]** :	(5/15)(3/6) = 1/6.

(A good example of the characteristics of the shrinking sample band in the case of overlapping sets, can be seen in the probability for outcome [f3, f5]. Picking f3 first causes f1, f2, f3 AND f4 to be removed from the band leaving f5 alone which will therefore be picked next (with probability of 1).)

Again, the total probability of getting any one of the list of possible solutions is unity.

A3.3 Example 3

The sets and weights in this example are identical to those of Example 2 but this time it is assumed that two modes can fire within set (A); the same execution strategy is adopted. The freedom to fire more than one mode in a set influences the shrinkage characteristics of the sample band in a similar manner to that observed in the case of intersecting mode sets.

In this example, there are 24 possible outcomes. These outcomes eventuate from the 6 permutations of each of the possible designated firings. The notation (f1,f2,f4) is used here to indicate, for example, that f1, f2 and f4 are sampled for firing but in no particular order. The firing solutions are given below along with associated probabilities:

(f1, f2, f4) : $(1/15)(2/14)(4/9) + (4/15)(1/3) + (4/15)(2/3) + (2/15)(4/13) + (2/15)(1/13)(4/9) + (1/15)(4/14)$

$= 4/945 + 4/45 + 8/45 + 8/195 + 8/1755 + 2/105$

$= 458/1365;$

(f1, f2, f5) : $(1/15)(2/14)(5/9) + (5/15)(1/6)(2/5) + (5/15)(2/6)(1/4) + (2/15)(5/13)(1/4) + (2/15)(1/13)(5/9) + (1/15)(5/14)(2/5)$

$= 1/189 + 1/45 + 1/36 + 1/78 + 2/351 + 1/105$

$= 1/12;$

(f1, f3, f5) : $(1/15)(3/14) + (5/15)(1/6)(3/5) + (5/15)(3/6)(1/3) + (3/15)(5/8)(1/3) + (3/15)(1/8) + (1/15)(5/14)(3/5)$

$= 1/70 + 1/30 + 1/18 + 1/24 + 1/40 + 1/70$

$= 58/315;$

(f2, f3, f5) : $(2/15)(3/13) + (5/15)(2/6)(3/4) + (5/15)(3/6)(2/3) + (3/15)(5/8)(2/3) + (3/15)(2/8) + (2/15)(5/13)(3/4)$

$= 2/65 + 1/12 + 1/9 + 1/12 + 1/20 + 1/26$

$= 929/2340;$

Again, the total probability of getting any one of the list of possible solutions is unity.

The above examples illustrate the steps taken to determine which modes should fire given the dependent mode sets, the mode weights and how many modes can fire within each set. In an EPN the number of modes that could fire within a given dependent mode set is a function of the input requirements of the associated modes

and the enabling marking. The Prolog implementation of the above strategy generates each of the possible solutions as multiple solutions to the firing problem and calculates an associated probability of firing.

An Explanation and Analysis Capability for Extended Petri Nets

Mike Davies, Fred D.J. Bowden, John M. Dunn

(DSTO-TR-0461)

DISTRIBUTION LIST

Number of Copies

AUSTRALIA

DEFENCE ORGANISATION

Task sponsor:

Director General Force Development (Joint) (Attention: DOIS)	1

S&T Program

Chief Defence Scientist)	
FAS Science Policy)	1 shared copy
AS Science Industry External Relations)	
AS Science Corporate Management)	
Counsellor, Defence Science, London	Doc Control sheet
Counsellor, Defence Science, Washington	Doc Control sheet
Scientific Adviser to MRDC Thailand	Doc Control sheet
Director General Scientific Advisers and Trials)	1 shared copy
Scientific Adviser - Policy and Command)	
Navy Scientific Adviser	3 copies of Doc Control sheet and 1 distribution list
Scientific Adviser - Army	Doc Control sheet and 1 distribution list
Air Force Scientific Adviser	1
Director Trials	1

Aeronautical & Maritime Research Laboratory

Director	1
Chief, Air Operations Division	1
Chief, Maritime Operations Division	1

Electronics and Surveillance Research Laboratory

Director	1
Chief Information Technology Division	1
Research Leader Command & Control and Intelligence Systems	1
Research Leader Military Computing Systems	1
Research Leader Command, Control and Communications	1
Executive Officer, Information Technology Division	Doc Control sheet
Head, Information Architectures Group	1
Head, C3I Systems Engineering Group	Doc Control sheet

Head, Information Warfare Studies Group	1
Head, Software Engineering Group	Doc Control sheet
Head, Trusted Computer Systems Group	Doc Control sheet
Head, Advanced Computer Capabilities Group	Doc Control sheet
Head, Computer Systems Architecture Group	Doc Control sheet
Head, Systems Simulation and Assessment Group	1
Head, Intelligence Systems Group	Doc Control sheet
Head Command Support Systems Group	1
Head, C3I Operational Analysis Group	1
Head Information Management and Fusion Group	1
Head, Human Systems Integration Group	Doc Control sheet
Authors	3
Publications and Publicity Officer, ITD	1
Chief, Land Space and Optoelectronics Division (Attention: Head, Operational Analysis)	1

DSTO Library and Archives

Library Fishermens Bend	1
Library Maribyrnong	1
Library DSTOS	2
Australian Archives	1
Library, MOD, Pyrmont	Doc Control sheet

Forces Executive

Director General Force Development (Sea),	Doc Control sheet
Director General Force Development (Land),	Doc Control sheet
Director General Force Development (Air),	Doc Control sheet

Navy

SO (Science), Director of Naval Warfare, Maritime Headquarters Annex
Garden Island, NSW 2000. 1

Army

ABCA Office, G-1-34, Russell Offices, Canberra 4
SO (Science), HQ 1 Division, Milpo, Enoggera, Qld 4057 NAPOC QWG Engineer
NBCD c/- DENGRS-A, HQ Engineer Centre
Liverpool Military Area, NSW 2174

S&I Program

Defence Intelligence Organisation	1
Library, Defence Signals Directorate	Doc Control sheet

B&M Program (libraries)

OIC TRS, Defence Central Library	1
Officer in Charge, Document Exchange Centre (DEC),	1
US Defence Technical Information Center,	2
UK Defence Research Information Centre,	2
Canada Defence Scientific Information Service,	1

NZ Defence Information Centre,	1
National Library of Australia,	1
Universities and Colleges	
Australian Defence Force Academy	1
Library	1
Head of Aerospace and Mechanical Engineering	1
Deakin University, Serials Section (M list)), Deakin University Library,	1
Senior Librarian, Hargrave Library, Monash University	1
Librarian, Flinders University	1
Librarian, University of Adelaide (Attention: Prof. Charles Pearce)	1
Librarian, University of South Australia (Attention: Prof. Jonathon Billington)	1
Librarian, La Trobe University (Attention: Arnold N. Pears)	1
Other Organisations	
NASA (Canberra)	1
AGPS	1
State Library of South Australia	1
Parliamentary Library, South Australia	1

OUTSIDE AUSTRALIA

Abstracting and Information Organisations	
INSPEC: Acquisitions Section Institution of Electrical Engineers	1
Library, Chemical Abstracts Reference Service	1
Engineering Societies Library, US	1
American Society for Metals	1
Documents Librarian, The Center for Research Libraries, US	1
Information Exchange Agreement Partners	
Acquisitions Unit, Science Reference and Information Service, UK	1
Library - Exchange Desk, National Institute of Standards and Technology, US	1
SPARES	10
Total number of copies:	**74**

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION
DOCUMENT CONTROL DATA

1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)

2. TITLE

An Explanation and Analysis Capability For Extended Petri Nets

3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)

Document (U)
Title (U)
Abstract (U)

4. AUTHOR(S)

Mike Davies, Fred D.J. Bowden and John M. Dunn

5. CORPORATE AUTHOR

Electronics and Surveillance Research Laboratory
PO Box 1500
Salisbury SA 5108

6a. DSTO NUMBER	6b. AR NUMBER	6c. TYPE OF REPORT	7. DOCUMENT DATE
DSTO-TR-0461	AR-009-951	Technical Report	December 1996

8. FILE NUMBER	9. TASK NUMBER	10. TASK SPONSOR	11. NO. OF PAGES	12. NO. OF REFERENCES
N9505/13/12	ADF 93/237	DGFD(J)	61	9

13. DOWNGRADING/DELIMITING INSTRUCTIONS
N/A

14. RELEASE AUTHORITY

Chief, Information Technology Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE CENTRE, DIS NETWORK OFFICE, DEPT OF DEFENCE, CAMPBELL PARK OFFICES, CANBERRA ACT 2600

16. DELIBERATE ANNOUNCEMENT

No limitations

17. CASUAL ANNOUNCEMENT Yes

18. DEFTEST DESCRIPTORS

Extended Petri nets; Command, Control, Communications and Intelligence

19. ABSTRACT

Extended Petri nets (EPN) have been adopted to help establish a number of capabilities for the study of military Command, Control, Communication and Intelligence (C3I). Regardless of the specific application, an EPN explanation and analysis capability is very important. This is particularly true where EPN have been used to describe artificial representations of real-world C3I entities. Some explanation and analysis capability is needed so that the underlying assumptions and characteristics of such an artificial representation can be clearly conveyed to, and judged by, an appropriate military domain expert. This document reports a general-purpose explanation and analysis capability for a class of EPN, that has been developed as part of the Distributed Interactive C3I Effectiveness (DICE) simulation activity within Information Technology Division.